BÁO CÁO CHƯƠNG 4

Cameras Models and Augmented Reality

1. The Pin-hole camera Model
   1. The camera matrix

* Camera metrix có thể được phân tách như sau:

A picture containing text

Description automatically generated

* Trong đó R là một ma trận xoay mô tả hướng của camera, t là một vector dịch chuyển 3D mô tả vị trí của camera center, và K là một intrinsic calibration matrix( Ma trận hiệu chuẩn nội tại) mô tả các thuộc tính chiếu của camera
* K chỉ phụ thuộc vào các thuộc tính của camera:

Text

Description automatically generated with low confidence

* Trong đó focal length(tiêu cự) f là khoảng cách giữa mặt phẳng ảnh và điểm camera center
* Skew(độ nghiêng) chỉ được sử dụng nếu mảng pixel trong cảm biến bị nghiêng và có thể trong hầu hết các trường hợp an toàn nó được đặt bằng 0 A picture containing text

  Description automatically generated. Khi đó chúng ta sẽ sử dụng fx, fy với fx = fy
* Aspect ratio, được sử dụng cho các phần tử pixel không vuông. Thông thường để an toàn thì giả sử =1, khi đó ma trận K trở thành Text

  Description automatically generated with low confidencecx, cy là toạ độ của điểm trung tâm camera, ta có thể xấp xỉ chúng với w/2, h/2 của ảnh. Ẩn chưa biết ở đây chỉ còn lại f
  1. Projecting 3D points
* Tạo một lớp camera chứa phương thức project để thực hiện phép chiếu lên mặt phẳng
* Đầu tiên sẽ tạo các điểm vào toạ độ đồng nhất và tạo ra đối tượng Camera với một ma trận chiếu trước khi thực hiện chiếu các điểm 3D và vẽ chúng. Kết quả như sau:

Chart, scatter chart

Description automatically generated

Hình 1 Vẽ các điểm chiếu từ 3D lên mặt phẳng qua phép chiếu

Shape, arrow

Description automatically generated

Hình 2 Hình chiếu sau khi góc máy camera được xoay quanh trục của vector a ngẫu nhiên

* 1. Factoring the camera matrix (Tính toán ma trận camera)
* Chúng ta cần khôi phục các thông số bên trong K và vị trị camera, và t, R. Việc phân vùng ma trận được gọi là factorization. Trong trường hợp này sẽ sử dụng factorization được gọi RQ-factozition.
* Bởi vì chúng ta cần ma trận R với định thức dường (nếu không thì toạ độ trục có thể bị lật) chúng ta có thể thêm một phép chuyển đổi T để thay đổi dấu khi cần
  1. Tính toán tâm camera
* Cho P là ma trận chiếu hay ma trận camera. C là một điểm 3D với thuộc tính PC=0.  và tâm camera có thể được tính không phụ thuộc vào ma trận K.

1. Camera Calibration ( Hiệu chỉnh máy ảnh)

Hiệu chỉnh máy ảnh là xác định các tham số bên trong camera, trong trường hợp chúng ra là ma trận K. Nó có thể mở rộng mô hình camera này để bao gồm méo xuyên tâm và các đồ tạo tác khác nếu máy tính của bạn cần độ đo chính xác.

**Một phương pháp hiệu chỉnh đơn giản**

* Hầu hết các tham số có thể thiết lập các giả thuyết cơ bản, phần khó khăn nhất là lấy được chính xác độ dài tiêu cự. Trong cái phương pháp hiệu chỉnh này bạn cần một đối tượng hiệu chỉnh hình chữ nhật phẳng( một quyển sách có thể làm được), thước đo hay thước kẻ và thích một bề mặt phẳng. Ở đây, các bước sẽ làm là:
  + Đo các bên của đối tượng hiệu chỉnh hình chữ nhật. Gọi là dX, dY
  + Đặt camera và đối tượng hiệu chỉnh trong một bề mặt phẳng đề camera quay lại và đối tượng hiệu chỉnh song song và gần như là nằm ở trung tâm của camera. Bạn sẽ phải nâng camera hoặc đối tượng để có một sự liên kết tốt
  + Đo khoảng cách từ máy ảnh tới đối tượng. Gọi là dZ
  + Chụp 1 bức ảnh và kiểm tra xem thiết lập đã thẳng hàng hay không, nghĩa là các cạnh của đối tượng hiệu chỉnh căn chỉnh so với hàng mà cột của hình ảnh
  + Đo chiều dài và chiều rộng của bức ảnh theo pixels. Gọi là dx, dy.
  + Bây giờ, tính tiêu cự như sau: A picture containing text

    Description automatically generated

1. Pose estimation from Planes and markers

* Trong chương 3 chúng ta đã biết các ước lượng đồng nhất giữa hai mặt phẳng. Kết hợp chúng với một máy ảnh được hiệu chỉnh thì có thể tính toán được tư thế của máy ảnh (rotation, tranformation) nếu hình ảnh chứa một đối tượng có thể chứa một đối tượng đánh dấu phẳng. Đây là đối tượng đánh dấu có thể được hầu như bất kì mặt phẳng nào
* Sử dụng SIFT để extract thuộc tính trong hai bức ảnh, và ước tính một phép đồng nhất sử dụng RANSAC
* Định nghĩa một hệ toạ độ 3D để các điểm đánh dấu nằm trong một mặt phẳng X-Y (Z = 0) với gốc nằm đâu đó trong điểm đánh dấu
* Để kiểm tra kết quả thì sử dụng một đối tượng 3D để đặt lên các điểm đánh dấu. Ở đây chúng ta sử dụng một khối lập phương và tạo các điểm lập phương.
* Với một phép đồng nhất và một ma trận hiệu chỉnh camera, chúng ta có thể xác định quan hệ dịch chuyển giữa 2 views
* Ở đây sử dụng phiên bản của hình ảnh với độ phân giải 747 \* 1000 và đầu tiên tạo ra một ma trận hiệu chỉnh chuẩn cho kích thước hình ảnh đó. Tiếp theo các điểm cho một hình lập phương tại điểm gốc được tạo. 5 điểm đầu tiên được tạo tương ứng với bottom. Sẽ nằm trên mặt phẳng Z = 0, mặt phẳng của điểm đánh dấu.
* Tạo ma trận camera đầu tiên với ma trận:

Icon

Description automatically generatedcó các trục toạ độ được căn chỉnh so với máy ảnh và đặt phía trên của điểm đánh dấu. 5 điểm 3D đầu tiên được chiếu vào hình ảnh. Với phép đồng nhất được ước tính có thể chuyển đổi chúng tới bức ảnh thứ 2. P2 = H P1, sẽ chuyển các điểm nằm trên điểm đánh dấu ở mặt phẳng Z=0 một cách chính xác. Điều này có nghĩa rằng hai cột đầu tiên và cột thứ 4 của ma trận P2 là chính xác. Bởi vì ta biết khối 3\*3 đầu tiên trong P2 là K\*R và R là một ma trận xoay, chúng ta có thể tính lại cột thứ 3 bằng cách nhận P2 với nghịch đảo của ma trận hiệu chỉnh K và thay thế cột thứ 3 với tích chéo của hai cột đầu tiên của R.

Calendar

Description automatically generated

A picture containing calendar

Description automatically generated

A picture containing calendar

Description automatically generated

1. Augmented Reality – Thực thể tăng cường

* Thực thể tăng cường là một thuật ngữ chung để đặt các đối tượng và thông tin lên một ảnh. Làm cho có cảm giác nó thuộc về ngữ cảnh và di chuyển tự nhiên trong trường hợp chuyển động của camera trong trường hợp
* Sử dụng hai thư viện PyGame và PyOpenGL để tạo một thực thể tăng cường
  1. From camera matrix to OpenGL format
* OpenGL sử dụng một ma trận 4\*4 để đại diện cho các phép dịch chuyển( cả dịch chuyển 3D và các phép chiếu). Khác với ma trận camera sử dụng 3 \* 4. Tuy nhiên, các phép dịch chuyển ngữ cảnh camera được tách biệt trong hai ma trận, GL\_PROJECTION matrix và GL\_MODELVIEW matrix. GL\_PROJECTION xử lý các thuộc tính chuẩn hoá ảnh và tương đương với ma trận hiệu chỉnh K. GL\_MODELVIEW xử lý dịch chuyển 3D của mối quan hệ giữa các đối tượng và camera. Cái này tương ứng với R và t trong matrix camera. Một điểm khác là hệ toạ độ được giả sử tại tâm của camera để GL\_MODELVIEW thực sự bao gồm phép dịch chuyển đặt một đối tượng trước camera.
* Hàm set\_projection\_from\_camera(K): Chúng ta giả sử sư hiệu chỉnh ở dạng đơn giản như 4.3 với trục quang học tại trung tâm của ảnh. Hàm glMatrixMode() thiết lập ma trận làm việc tới GL\_PROJECTION và các lệnh tiếp theo đây sẽ thay đổi nó. glLoadIdentity() reset bất kì thay đổi nào trước đó tới ma trận. Chúng ta tính toán chiều dọc của view theo độ với chiều cao của ảnh và độ dài tiêu cự cũng như là aspect ratio. Một OpenGL projection cũng có một mặt phẳng cắt gần và xa để giới hạn độ sâu của cái được xuất ra. Chúng ra sử dụng hàm gluPerspective() để thiết lập ma trận chiếu và định nghĩa và xác định toàn bộ hình ảnh là cổng xem( về cơ bản là những gì được hiển thị).
* Ma trận model view mã hoá độ xoay và dịch chuyển để đặt đối tượng trước camera. Nó là một ma trận 4 \* 4 trông như sau: A picture containing text, clock

  Description automatically generated. R là ma trận xoay với các cột là hướng của 3 trục toạ độ, t là vector dịch. Khi tạo phần xoay của ma trận model view thì sẽ cần để tổ chức tất cả các hướng( đối tượng và hệ toạ độ) bằng cách nhân các thành phần riêng cùng nhau. Đầu tiên chúng ta chuyển để làm việc với GL\_MODELVIEW và reset nó. Sau đó chúng tôi tạo một ma trận xoay 90 độ bởi vì đối tượng chúng ta muốn đặt cần được xoay. Sau đó để đảm bảo rằng phần xoay của ma trận camera thực sự là ma trận xoay trong trường hợp có lỗi hoặc là nhiễu khi chúng tôi ước tính ma trận camera. Thực hiện điều này với SVD và ma trậ xoay sấp sỉ tốt nhất là R = . Hệ trục trên OpenGL hơi khác nên chúng ta cần phải lật trục x. Sau đó chúng ta tạo M (model view matrix) bằng các nhân các ma trận xoay. Hàm glLoadMatrixf() thiết lập M và nhận một mảng 16 giá trị của ma trận M theo cột.
  1. Placing virtual objects in the image (Đặt các đối tượng ảo trong ảnh)
* Vẽ background bằng hàm draw\_background(imname)
* Vẽ teapot bằng hàm draw\_teapot(size)
  1. Typing it all together